Condition and Looping Statement

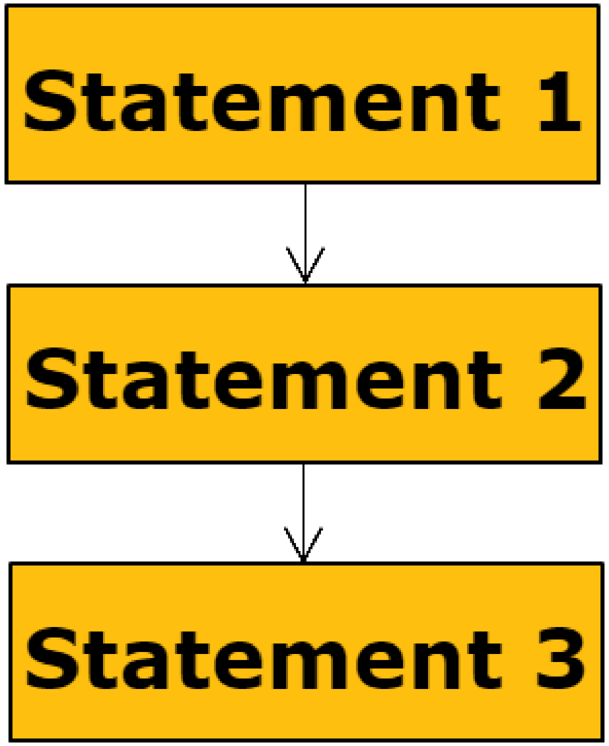
Objective

- About Statement Flow control.
- Condition Statements
  - if statement
  - if-else statement
  - elif statement
- Looping Statements
  - for loop
  - while loop
- Jump Statement
  - break
  - continue

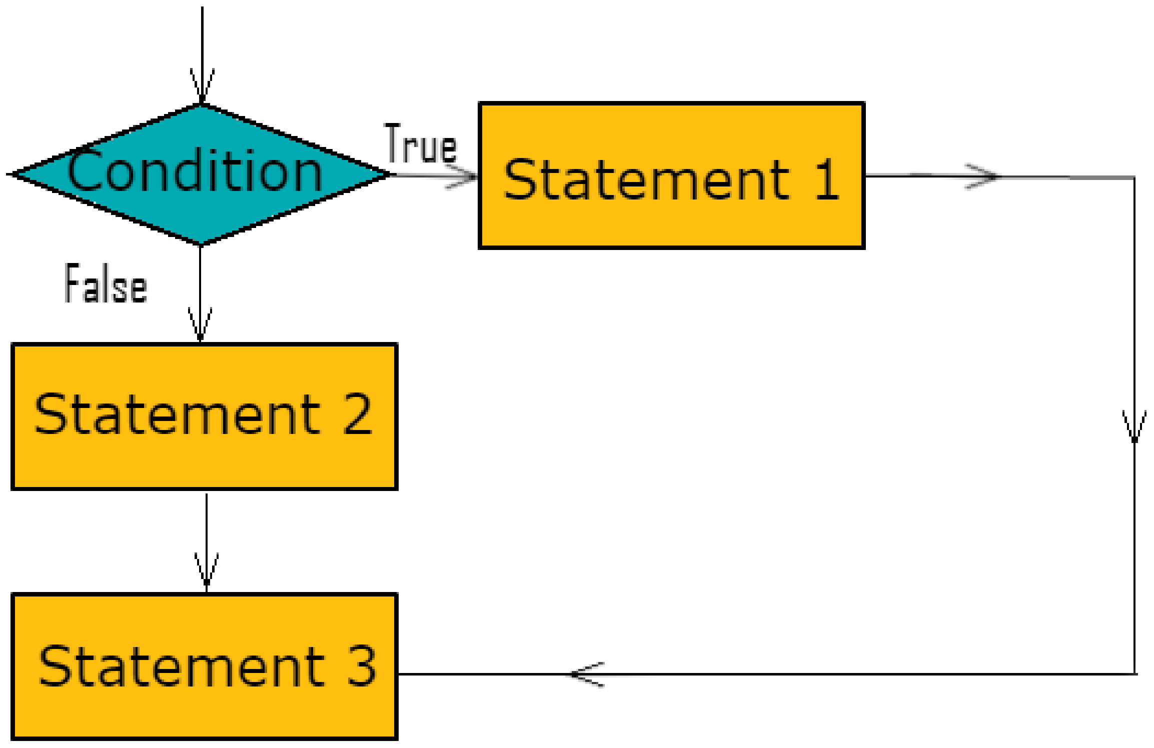
Statement Flow Control

- Sequence
- Selection/Condition
- Iteration / looping

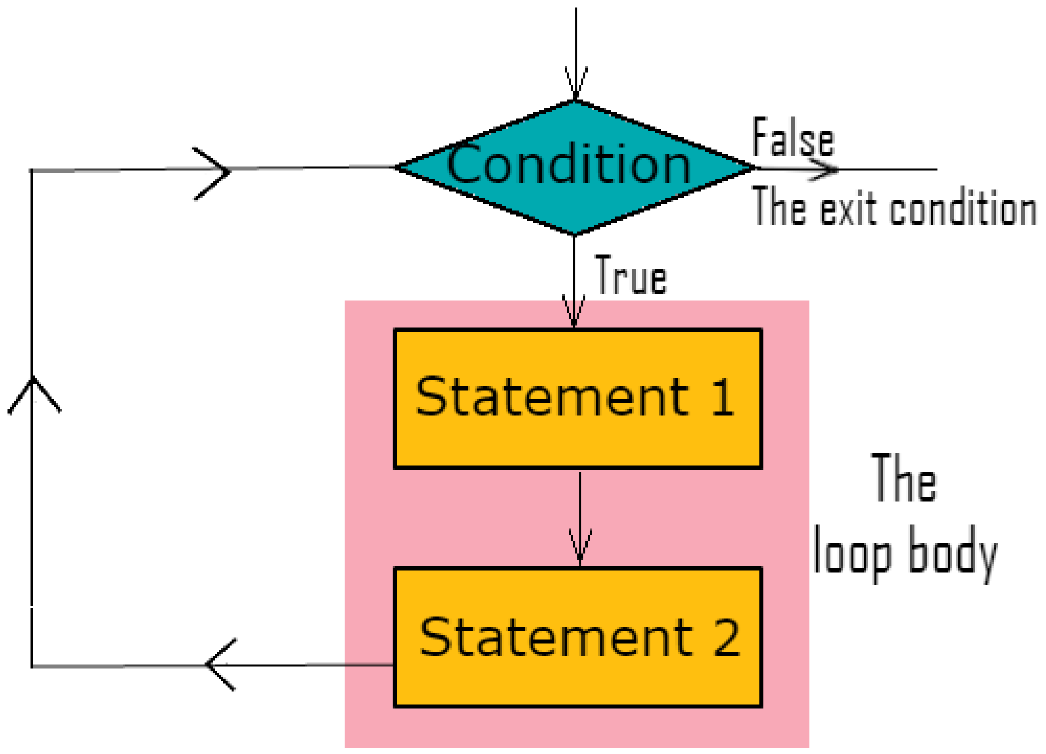
Sequence



Condition



Looping


Condition
False
The exit condition
True
Statement 1
Statement 2
The
loop body

if, elif, else statement

- An ***if Statement*** tests a particular condition, only if the condition is satisfied than the course-of-action is followed.
- An ***if-else Statement*** tests a particular condition, if the condition is satisfied than a certain course-of-action is followed, else another mentioned course-of-action has to be followed.
- *THE AFORE METIONED STATEMENTS CAN BE UNDESTOOD BY THEIR <u>GRAMATICAL USE OF if & else.</u>*

- An ***if-elif Statement*** tests a particular condition, if the condition is satisfied than the course-of-action is followed if not another condition is tested by elif statement.

if Statement: Syntax & Example

SYNTAX:

```
if  <conditional expression>:
          statement
```

EXAMPLE

CODE:

```
if 10 > 5:
    print "10 > 5"

if 5 > 10:
    print "5 > 10"
```

OUTPUT:

```
*** Remote Interpreter Reinitialized  ***
>>>
10 > 5
>>>
```

If-else Statement: Syntax & Example

SYNTAX:

```
if  <conditional expression>:
        statement1
else:
        statement2
```

EXAMPLE

CODE:

```
if 5 > 6:
        print “5 is greater than 6"
else:
        print “6 is greater than 5”
```

OUTPUT:

```
*** Remote Interpreter Reinitialized  ***
>>>
6 is greater than 5
>>>
```

If-elif Statement: Syntax

```
if  <conditional expression0>:
       statement 1
elif <conditional expression1>:
       statement 2
elif <conditional expression2>:
       statement 3
.   .   .   .   .   .   .
.   .   .   .   .   .   .
.   .   .   .   .   .   .
.   .   .   .   .   .   .
.   .   .   .   .   .   .
.   .   .   .   .   .   .
elif <conditional expression n>:
       statement n
else:
       statement n+1
```

If-elif Statement: Example

CODE:

```
a,b,c=1,2,5
if a==5:
    print "a is 5"
elif b==5:
    print "b is 5"
elif c==5:
    print "c is 5"
else:
    print "None of a,b,c is 5"
```

OUTPUT:

```
*** Remote Interpreter Reinitialized  ***
>>>
c is 5
>>>
```

Looping

‘for’ Statement:

The *‘for’ loop* of python is designed to process the items of any sequence, such as a list.

A *for loop* is processed as:

- A loop variable is assigned the first value in the sequence.
- All statements in the body of *for loop* are executed with assigned value of loop variable.
- The loop-variable is given a different value and again all the above mention process takes place.
- This continues until all values in the sequence are processed.

for Statement: Syntax & Example

SYNTAX:

```
for <variable> in <sequence>:
  statement_to_repeat
```

Example

CODE:

```
for a in [2,4,8]:
        print a*"*"
```

OUTPUT:

```
*** Remote Interpreter Reinitialized  ***
>>>
**
****
********
>>>
```

What if the sequence used in the *for loop* is too long ?

range() Function

The range() function has two sets of parameters, as follows:

- range(stop)

  stop: Number of whole numbers to generate, starting from zero. eg. range(3) == [0, 1, 2].

- range(start, stop, step)

  start: Starting number of the sequence.

  stop: Generate numbers up to, but not including this number.

  step: Difference between each number in the sequence.

for Statement using range() function

CODE:

```
for a in range (0,100,20):
    print a
```

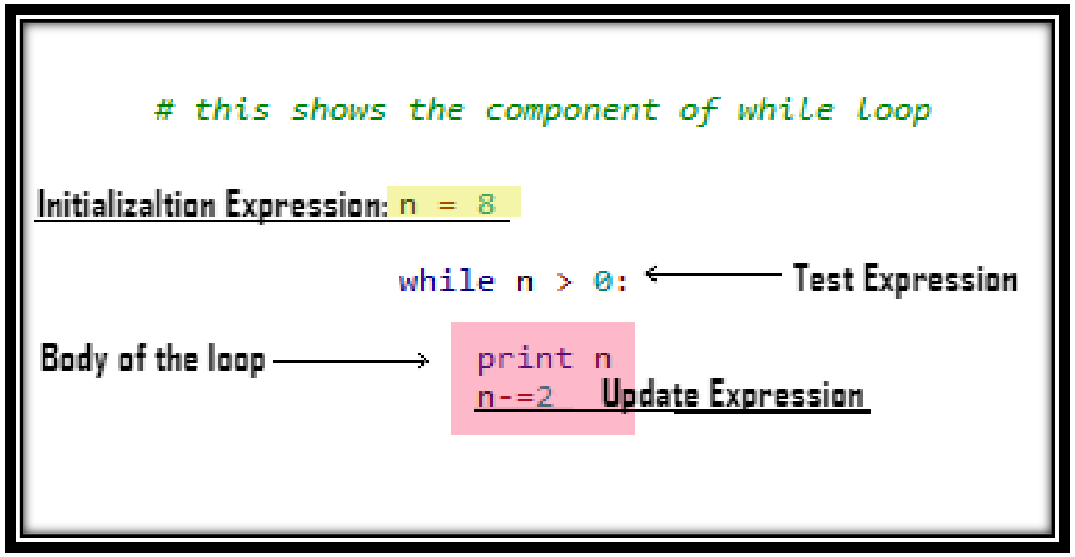
OUTPUT:

```
*** Remote Interpreter Reinitialized  ***
>>>
0
20
40
60
80
>>>
```

while Statement:

Any while loop has the following four elements:

1. ***Initialization Expression*** initializes the loop variable.
2. ***Test Expression*** decides weather the loop-body will be executed or not.
3. ***The body of loop*** are the statement that get repeated.
4. ***Update Expression*** changes the value of loop variable.



```
# this shows the component of while loop

n = 8

while n > 0:

    print n
    n-=2
```

while Statement: Syntax & Example

SYNTAX:

```
While <logical expression>:
      loop-body
```

Example

CODE:

```
a = 10
while a > 0:
    print "Hello",a
    a-=4
print"Loop Over"
```

OUTPUT:

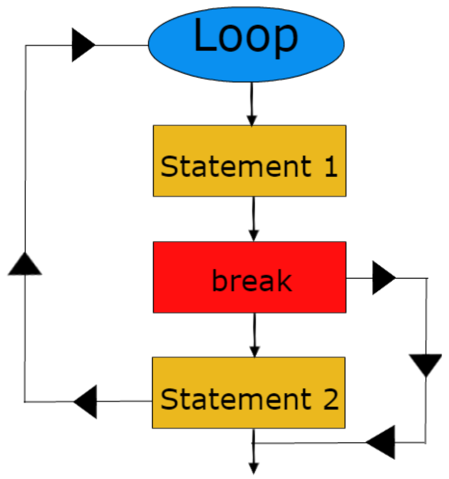
```
*** Remote Interpreter Reinitialized  ***
>>>
Hello 10
Hello 6
Hello 2
Loop Over
>>>
```

Jump Statement :

These statements enables program to skip over a part of program when needed.

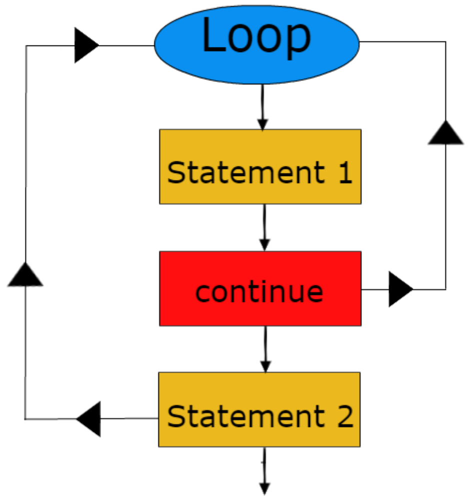
The break Statement

- The break statement terminates the very loop it lies within.
- That loop is not repeated once the break statement is executed.



The continue Statement

- The continue statement only terminates one cycle of the loop it lies within.
- The loop may start after the continue command is executed.


Loop
Statement 1
continue
Statement 2